

# Математика в Python

2018

# План

## Введение

## Диаграммы и графики

- Векторизация функций

- Двумерные графики

- Графики в изометрии

- Интерактивные графики

## Линейная алгебра

## Полиномы

## Численные методы

- Минимизация функции

- Интерполяция

## Теория вероятностей и статистика

- Гистограмма

- Корреляция и диаграмма рассеивания

- Диаграмма размаха

## Ссылки и литература

# Outline

## Введение

### Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

### Линейная алгебра

### Полиномы

### Численные методы

Минимизация функции

Интерполяция

### Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

### Ссылки и литература

```
In [7]: from math import *
import numpy as np
from matplotlib.pyplot import *
```

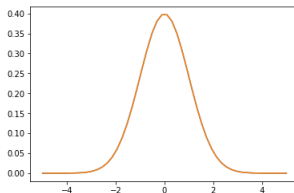
```
In [6]: math.sin( math.pi )
```

```
Out[6]: 1.2246467991473532e-16
```

```
In [14]: def norm_pdf(x):
return 1/sqrt(2*pi)*exp(-x**2/2)
```

```
X = np.linspace(-5, 5)
Y = [ norm_pdf(x) for x in X ]
```

```
plot(X,Y)
show()
```

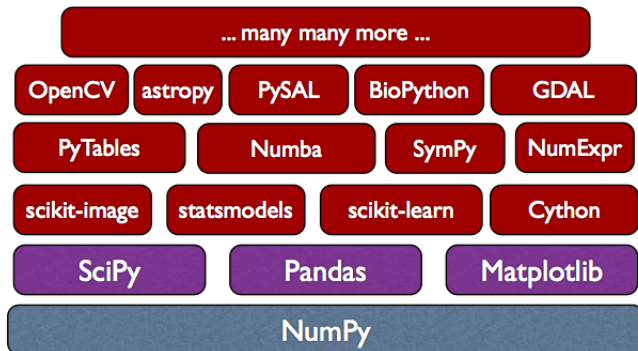


# Некоторые популярные математические пакеты

- ▶ **numpy** - работа с матрицами и многомерными массивами
- ▶ **scipy** - научные и инженерные вычисления, использует numpy;
- ▶ **matplotlib** - построение графиков и диаграмм;
- ▶ **seaborn** - визуализация статистических данных, эстетичнее чем matplotlib;
- ▶ **mpld3** - использование D3.js для построения интерактивных matplotlib графиков в окне браузера;
- ▶ **pandas** - анализ данных: статистики, регрессия, визуализация и т.п.

# Популярные математические пакеты

Многие пакеты построены на основе NumPy



# Outline

## Введение

## Диаграммы и графики

- Векторизация функций

- Двумерные графики

- Графики в изометрии

- Интерактивные графики

## Линейная алгебра

## Полиномы

## Численные методы

- Минимизация функции

- Интерполяция

## Теория вероятностей и статистика

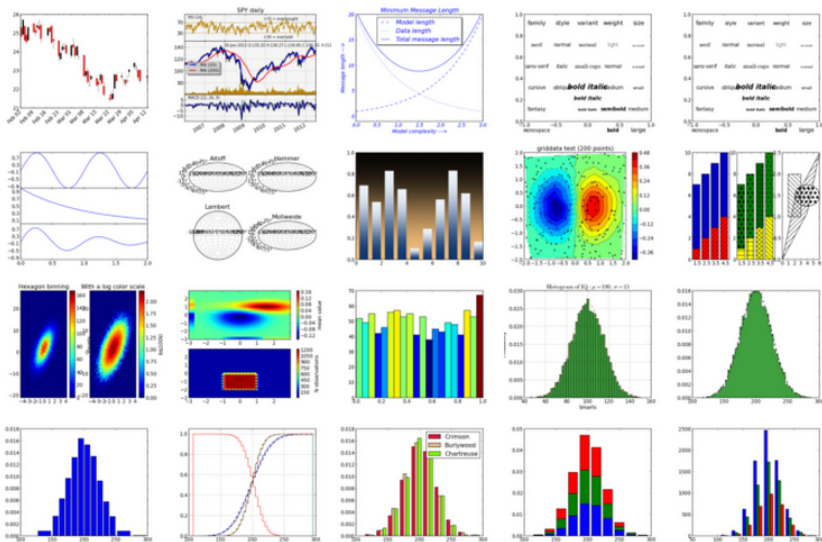
- Гистограмма

- Корреляция и диаграмма рассеивания

- Диаграмма размаха

## Ссылки и литература

# Пакет matplotlib





# Пакет matplotlib

Пакет **matplotlib** содержит модули предназначенные для построения диаграмм и графиков.

**Модуль pyplot** из этого пакета предназначен непосредственно для построения графиков. Этого модуля будет достаточно для построения относительно простых графиков.

Остальные модули пакета matplotlib содержат преимущественно функции для гибкой настройки вида графиков, осей, подписей к осям, компоновки нескольких графиков на одном листе и т.п.

# Пакет matplotlib

```
from matplotlib.pyplot import plot
```

Основные функции модуля pyplot

Создание графика **plot** - функция с переменным числом параметров. Если некоторые параметры не указаны, то им задаётся значение по-умолчанию.

- ▶ **plot(y-list)** - создаёт график в памяти программы  
y-list - список ординат (значений  $y$ ).  
В качестве абсцисс (значений  $x$ ) используются номера значений из списка y-list, т.е. индексы.

```
help(plot)
```

- ▶ **plot(x-list, y-list)**

x-list - список абсцисс (значений  $x$ ).

y-list - список ординат (значений  $y$ ).

Длины списков x-list и y-list должны быть одинаковы.

# Пакет matplotlib

## Дополнительные параметры

### ▶ **plot(x-list, y-list, style)**

style - стиль графика. Определяет цвет, вид кривой и точек

- ▶ - прямая линия
- ▶ - - пунктирная линия
- ▶ . только точки
- ▶ v треугольники вместо точек

## Цвета

- ▶ 'b' blue
- ▶ 'g' green
- ▶ 'r' red
- ▶ 'k' black

Дополнительную информацию о стилях см. в документации

# Пакет matplotlib

## Дополнительные параметры

- ▶ **plot(x-list, y-list, style, label)**  
label - подпись к графику. По-умолчанию не показывается.
- ▶ **plot(x-list, y-list, style, label, linewidth)**  
linewidth - толщина линии. По-умолчанию - 1.

# Пакет matplotlib

```
from matplotlib.pyplot import grid, xlabel, ylabel, legend
```

- ▶ **grid(True)** - "включает" координатную сетку.  
Шаг сетки выбирается автоматически.
- ▶ **xlabel("подпись")** - добавляет название для оси x
- ▶ **ylabel("подпись")** - добавляет название для оси y
- ▶ **legend(loc)** - добавляет к графикам пояснения (легенду)  
loc - положение блока с пояснениями. По-умолчанию - справа сверху.  
Для автоматического выбора положения следует задать параметр `loc = 'best'`

# Пакет matplotlib

```
from matplotlib.pyplot import show, savefig
```

- ▶ **show()** - создаёт и показывает окно содержащие построенные функциями `plot()` графики.  
Оси координат строятся автоматически, масштаб выбирается в зависимости от ширины и высоты графика.
- ▶ **savefig(filename [, dpi])** - сохраняет изображение графика в файл. Графический формат определяется по расширению файла.  
filename - имя файла  
dpi - количество точек на дюйм (DPI)

# Пакет matplotlib

Типичный алгоритм построения графика

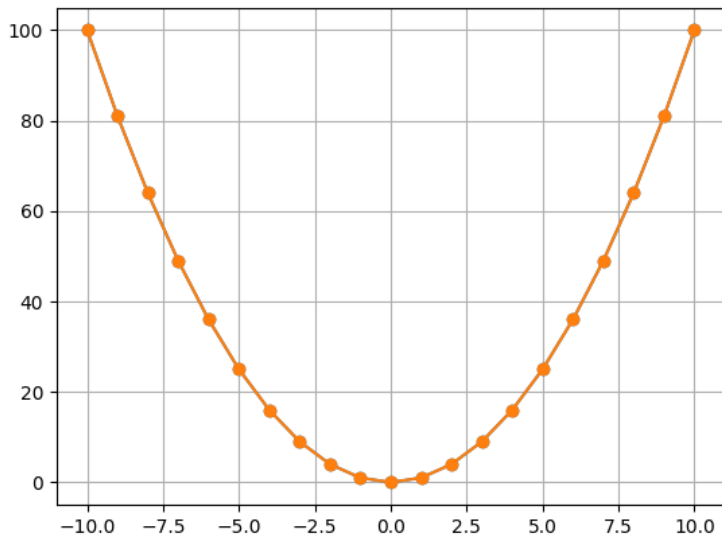
1. Создать графики
2. Настроить оформление графика
3. Показать или сохранить график



## Пакет matplotlib

```
X = list(range(-10,11))  
Y = [x**2 for x in X]  
plot(X,Y, '-o')  
grid(True)  
show()
```

## Пакет matplotlib

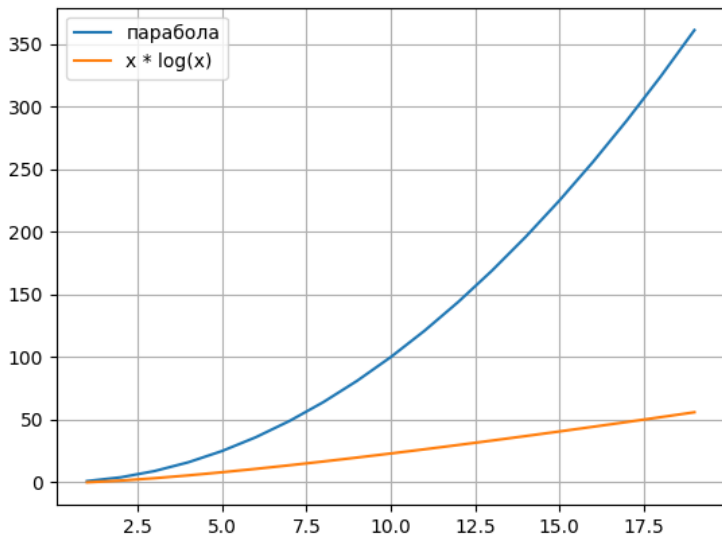


## Пакет matplotlib

На одном поле можно построить несколько графиков. Чтобы их различать стоит указать для них названия. Для этого будем явно задавать имя параметра: **label**

```
X = list(range(20))
Y1 = [x**2 for x in X]
Y2 = [x * log(x) for x in X]
plot(X,Y, label = 'парабола')
plot(X,Y, label = 'x * log(x)')
grid(True)
show()
```

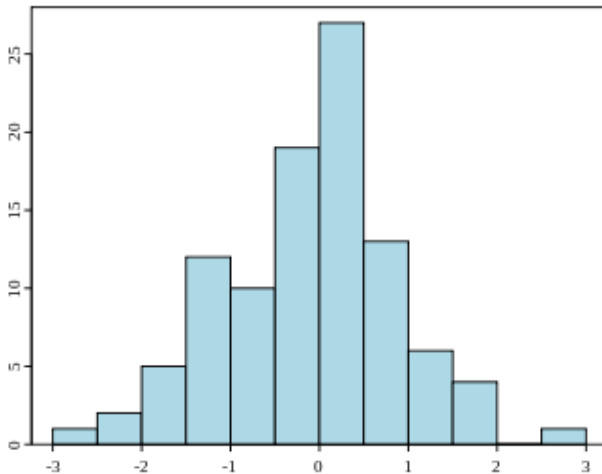
## Пакет matplotlib



# Пакет matplotlib

## Гистограммы

Гистограмма - столбчатая диаграмма, способ графического представления табличных данных



# Пакет matplotlib

```
from matplotlib.pyplot import bar
```

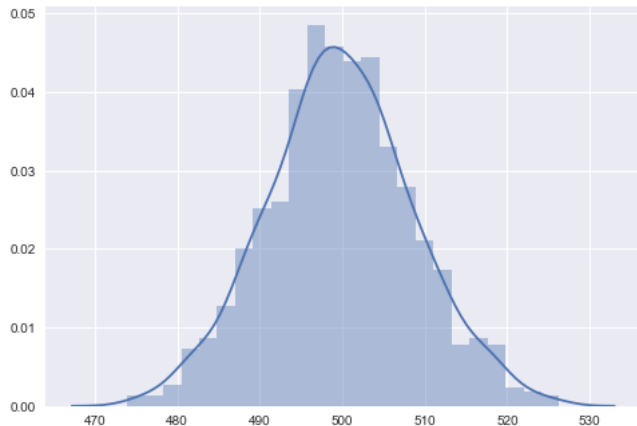
**bar(x-values, y-values)** - строит гистограмму.

x-values - значения x

y-values - значения y (высоты столбцов)

## Пакет matplotlib

```
X = [2,3,4,5]  
Y = [10,20, 15,5]  
bar(X,Y)  
show()
```



## Добавление дополнительных элементов на график

- ▶ Заголовок

```
title('1a TITLE')
```

- ▶ Текст

```
text(x,y, 'текст')
```



# Пакет matplotlib

## Matplotlib и Jupyter

По-умолчанию графики построенные в Jupyter будут показаны в ячейке вывода.

Чтобы показать их в отдельном окне, с возможностью масштабирования и перемещения следует перед построением графиков выполнить в Jupyter команду:

```
%matplotlib
```

Чтобы вернуть построение графиков в блокнот нужно выполнить команду

```
%matplotlib inline
```

## Эстетика графиков

На основе matplotlib создано много пакетов для визуализации данных. Один из них - **sebaorn**. Этот пакет предназначен для представления статистических данных и поэтому работа с ним может показаться немного сложнее.

Графики построенные этим модулем смотрятся лучше, чем аналогичные, построенные с помощью matplotlib (со стандартными настройками)

Для того чтобы использовать стиль оформления seaborn достаточно просто подключить этот модуль и далее работать с matplotlib.

# Пакет matplotlib

## Эстетика графиков

На основе matplotlib создано много пакетов для визуализации данных. Один из них - **sebaorn**. Этот пакет предназначен для представления статистических данных и поэтому работа с ним может показаться немного сложнее.

Графики построенные этим модулем могут выглядеть лучше, чем аналогичные, построенные с помощью matplotlib (со стандартными настройками)

Для того чтобы использовать стиль оформления seaborn достаточно просто подключить этот модуль и далее работать с matplotlib.

```
import seaborn
```

## Пакет matplotlib

Пакет seaborn не входит в стандартную библиотеку Python. Поэтому его придётся установить отдельно используя программу pip (поставляется вместе с Python)

```
pip.exe install seaborn
```

# Пакет matplotlib

Ещё примеры и документация:

[matplotlib.org/tutorials/introductory/pyplot.html](https://matplotlib.org/tutorials/introductory/pyplot.html)

# Outline

Введение

Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

Линейная алгебра

Полиномы

Численные методы

Минимизация функции

Интерполяция

Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

Ссылки и литература

## Векторизация функций

Для построение трёхмерных графиков неудобно пользоваться скалярными функциями, вручную вычисляя значения для каждого значения аргумента.

Класс **vectorize** пакета `numpy` может быть использован для "векторизации" функции. Такая функция может принимать в качестве параметра `array` и возвращать `array`.

```
from math import *
import numpy as np

def foo(x):
    return x*sin(x)

foo = np.vectorize(foo)
```

# Векторизация функций

```
from math import *
from matplotlib.pyplot import *
import numpy as np

def foo(x):
    return x*sin(x)

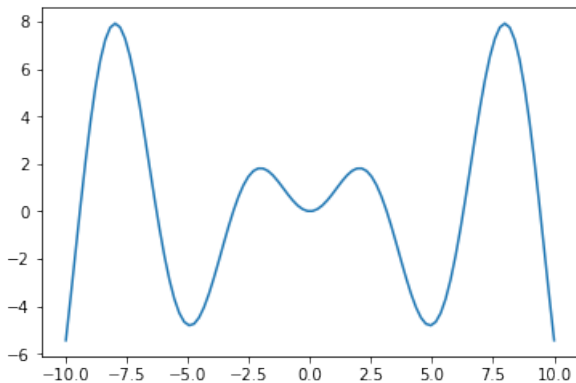
foo = np.vectorize(foo)

# создание array из 100 точек отрезка [-10, 10]
x = np.linspace(-10, 10, 100)
y = foo(x)

plot(x,y)
show()
```



# Векторизация функций



# Векторизация функций

Вместо обычных функций можно использовать  
лямбда-функции

```
from math import *  
import numpy as np  
  
foo = np.vectorize(lambda x: x*sin(x))
```

# Outline

Введение

Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

Линейная алгебра

Полиномы

Численные методы

Минимизация функции

Интерполяция

Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

Ссылки и литература

## Пример

Тепловая карта с линиями уровня

```
from matplotlib.pyplot import *
import numpy as np
from math import *

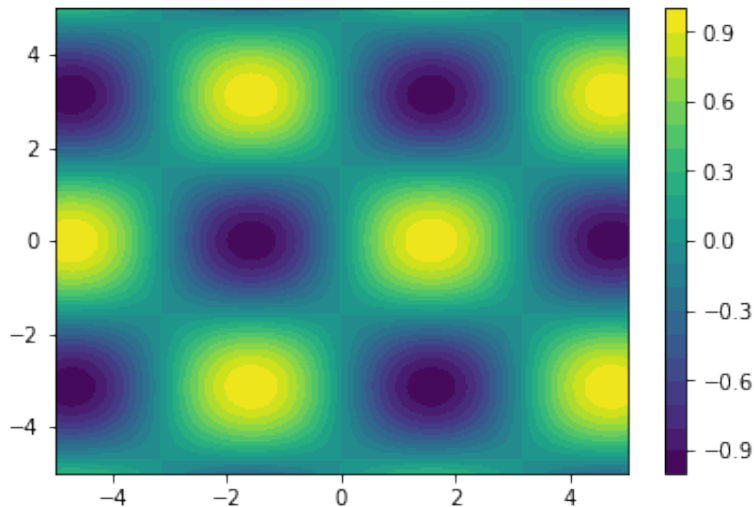
X = np.linspace(-5, 5, 100)
Y = np.linspace(-5, 5, 100)

# создание сетки (декартового произведения X и Y)
xx,yy = np.meshgrid( X, Y)

foo = np.vectorize(lambda x,y: sin(x)*cos(y))
zz = foo(xx,yy)

contourf(xx, yy, zz, 20)
colorbar()
show()
```

## Пример



Ещё примеры:

[jakevdp.github.io/PythonDataScienceHandbook/04.04-density-and-contour-plots.html](https://jakevdp.github.io/PythonDataScienceHandbook/04.04-density-and-contour-plots.html)

# Outline

## Введение

## Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

## Линейная алгебра

## Полиномы

## Численные методы

Минимизация функции

Интерполяция

## Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

## Ссылки и литература

## Графики в изометрии

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

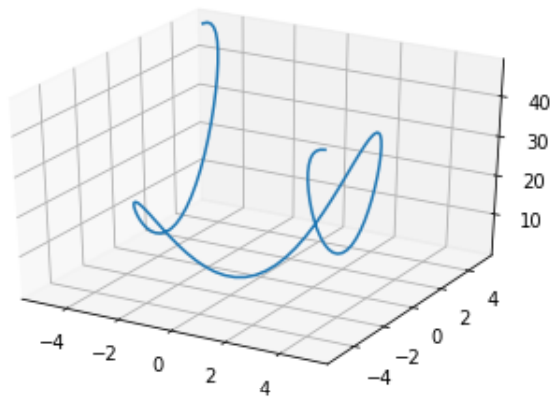
fig = plt.figure()
ax = fig.gca(projection='3d')

x = np.linspace(-5, 5, 100)
y = np.sin(x)*5
z = x**2 + y**2

ax.plot(x, y, z, antialiased=True)

plt.show()
```





## Графики в изометрии

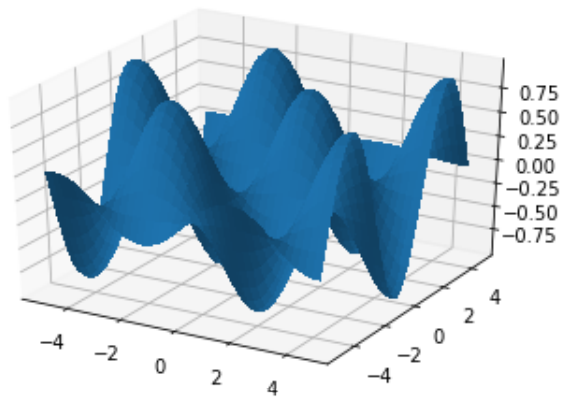
```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
from math import *
```

```
fig = plt.figure()
ax = fig.gca(projection='3d')
```

```
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
```

```
foo = np.vectorize(lambda x, y: sin(x) * cos(y))
Z = foo(X,Y)
```

```
ax.plot_surface(X, Y, Z, antialiased=False)
plt.show()
```



# Графики в изометрии

Дополнительно можно раскрасить поверхность как тепловую карту, изобразить сетку вместо поверхности и настроить другие параметры отображения.

Документации matplotlib с примерами:  
[matplotlib.org/mpl\\_toolkits/mplot3d/tutorial.html](https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html)

# Outline

## Введение

## Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

## Линейная алгебра

## Полиномы

## Численные методы

Минимизация функции

Интерполяция

## Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

## Ссылки и литература

plot.ly



plotly - модуль для создание интерактивных графиков.

Существуют версии для Python, R, Matlab

Графики представляют собой html фрейм с JavaScript (фреймворк D3.js).

Существуют два режима редима с графиками:

- ▶ оффлайн (график сохраняется как отдельный html файл)
- ▶ графики публикуются на сервере plot.ly

Для создания оффлайн-графиков нужно использовать отдельный модуль offline пакета plotly.

```
from plotly.offline import download_plotlyjs, \
    init_notebook_mode, plot, iplot
from plotly.graph_objs import Scatter, Figure, Layout
import numpy as np
```

```
x = np.linspace(-10, 10, 100)
y = np.vectorize(lambda x: np.sin(x) * x)(x)
plot([Scatter(x=x, y=y)])
```

Будет создан отдельный html файл, который автоматически откроется вбраузере



## plotly

Чтобы графики отображались в окне Jupyter следует явно это указать

```
init_notebook_mode(connected=True)
```

и использовать функцию рисовани **iplot** вместо **plot**.

```
from plotly.offline import download_plotlyjs, \
    init_notebook_mode, plot, iplot
from plotly.graph_objs import Scatter, Figure, Layout
import numpy as np
```

```
init_notebook_mode(connected=True)
```

```
x = np.linspace(-10, 10, 100)
y = np.vectorize(lambda x: np.sin(x) * x)(x)
iplot([Scatter(x=x, y=y)])
```

## plotly

Диаграмма рассеивания трёхмерной случайной величины

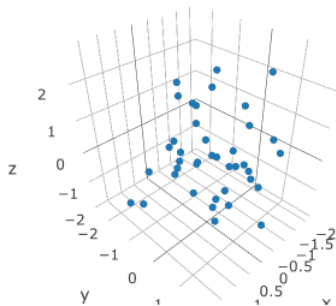
```
from plotly.offline import download_plotlyjs, \
    init_notebook_mode, iplot
from plotly.graph_objs import Scatter3d, Figure
import numpy as np

init_notebook_mode(connected=True)
# создание 3-х массивов заполненных 10 случайными значениями
x,y,z = np.random.uniform( size=(3, 10))

trace1 = Scatter3d(x=x, y=y, z=z, mode='markers', \
    marker=dict(size=2))

data = [trace1]
fig = Figure(data=data)
iplot(fig)
```

# Пример



# Outline

Введение

Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

Линейная алгебра

Полиномы

Численные методы

Минимизация функции

Интерполяция

Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

Ссылки и литература

# Линейная алгебра

```
import numpy as np

# Создание матрицы
A = np.array([ [1,2,3],
               [3,2,1],
               [2,1,3] ] )
```

Матрица A - это двумерный массив (список), только в обёртке numpy.

# Линейная алгебра

*# умножение на число*

```
A * 3.14
```

```
> array([[ 3.14,  6.28,  9.42],  
        [ 9.42,  6.28,  3.14],  
        [ 6.28,  3.14,  9.42]])
```

*# Умножение на вектор*

```
A * [2,3,4]
```

```
array([[ 2,  6, 12],  
       [ 6,  6,  4],  
       [ 4,  3, 12]])
```

Причём здесь вектор не обязательно должен быть типом numpy.

# Линейная алгебра

*# Сложение матриц*

$A + B$

*# Умножение матриц*

$A @ B$

*# или*

`np.dot(A,B)`

*# Обратная матрица*

`np.linalg.inv(A)`

# Линейная алгебра

## Решение СЛАУ

*# Настройка вывода. Число знаков после запятой - 4.*

*# не выводить числа в экспоненциальной форме*

```
numpy.set_printoptions(precision=4, suppress=True)
```

```
A = np.matrix([
```

```
    [1, 2, 3],
```

```
    [3, 2, 1],
```

```
    [2, 3, 1]])
```

```
B = np.matrix([[1], [2], [3]])
```

```
X = np.linalg.solve(A, B)
```

```
matrix([[ 0.0833],
```

```
        [ 1.0833],
```

```
        [-0.4167]])
```

Стоит обратить внимание на то, что вектор-столбец определяется как матрица из одного столбца, а не как список.



# Outline

Введение

Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

Линейная алгебра

**Полиномы**

Численные методы

Минимизация функции

Интерполяция

Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

Ссылки и литература

# Полиномы

```
import numpy as np
# полином одной переменной
p1 = np.poly1d([2.1, -0.3, -2])

print(p1)
# 2.1 x^2 - 0.3 x + 7

# вычисление значения полинома
p1(10) # 214.0

# корни полинома (p1 = 0)
p1.r
# [ 1.04993917 -0.90708203 ]
```

# Полиномы

```
from matplotlib.pyplot import *
```

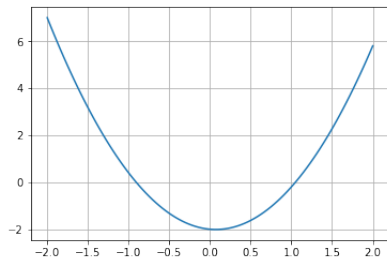
```
x = np.linspace(-2, 2, 100)
```

```
y = [p1(x) for x in x]
```

```
plot(x,y)
```

```
grid(True)
```

```
show()
```



# Outline

Введение

Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

Линейная алгебра

Полиномы

Численные методы

Минимизация функции

Интерполяция

Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

Ссылки и литература

# Outline

## Введение

## Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

## Линейная алгебра

## Полиномы

## Численные методы

Минимизация функции

Интерполяция

## Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

## Ссылки и литература

# Минимизация функции

```
from scipy.optimize import minimize
x0 = 100 # начальное значение
minimize(lambda x: return (x-3)*(x-3) - 5, x0)
```

Вывод:

```
nfev: 12
jac: array([ 0.])
message: 'Optimization terminated successfully.'
fun: -5.0
success: True
x: array([ 3.])
njev: 4
hess_inv: array([[ 0.5]])
status: 0
```

Минимум функции:  $f(3) = -5$

# Outline

## Введение

## Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

## Линейная алгебра

## Полиномы

## Численные методы

Минимизация функции

Интерполяция

## Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

## Ссылки и литература

# Численные методы

## Интерполяция

**Интерполяция** - нахождение промежуточных значений величины по имеющемуся дискретному набору известных значений.



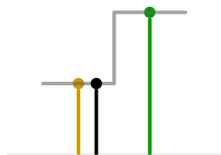
**Слайн** (spline) — функция, область определения которой разбита на конечное число отрезков, на каждом из которых она совпадает с некоторым алгебраическим многочленом (полиномом).

Максимальная из степеней использованных полиномов называется **степенью сплайна**.

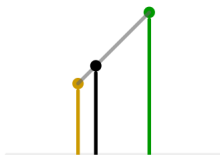
сплайн — это кусочно заданная функция

# Слайн

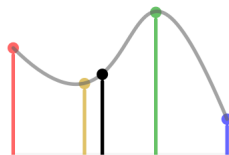
Примеры интерполяции для отдельных участков.



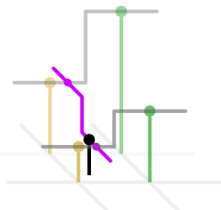
1D nearest-neighbour



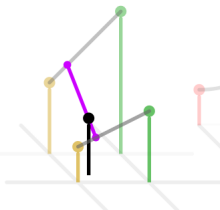
Linear



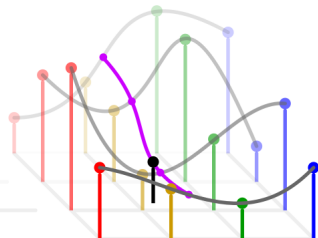
Cubic



2D nearest-neighbour



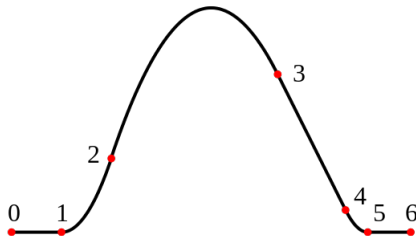
Bilinear



Bicubic

# График кучочно заданной функции

Сплайн второй степени



- ▶ 0-1 Прямая
- ▶ 1-2 Парабола
- ▶ 2-3 Парабола
- ▶ 3-4 Прямая
- ▶ 4-5 Парабола
- ▶ 5-6 Прямая

# Интерполяция

Интерполяция функции одной переменной

```
from scipy.interpolate import interp1d
```

Возможна интерполяция сплайном первой, второй и третьей степени<sup>1</sup>

- ▶ linear
- ▶ quadratic
- ▶ cubic

---

<sup>1</sup>см. другие способы в документации

# Интерполяция

Пусть функция задана в табличном виде - набором значения  $X$  и  $Y$ .

Задача - определить значение функции для  $X$ , не перечисленного в таблице.

Применем для этого интерполяцию, построив функцию в аналитическом виде, которая будет проходить наиболее близко к заданным точкам.

## Пример 1

```
from scipy.interpolate import interp1d
```

```
X = [1,2,3,4,5,6,7]
```

```
Y = [1,4,9,16,25,36,49]
```

```
func = interp1d(X, Y, kind='cubic')
```

func - функция определённая на отрезке от  $\min(x)$  до  $\max(x)$ .

kind - тип интерполяции (см. справку)

В примере использована интерполяция сплайном третьего порядка.

Теперь можно вычислять значение функции в любой точке.

```
func(5.3)
```

```
# array(28.090000000000003)
```

## Пример 2

```
# таблично заданная функция
X = [1,2, 3, 8, 9, 12]
Y = [1,2,2.5, 20, 42, 30]

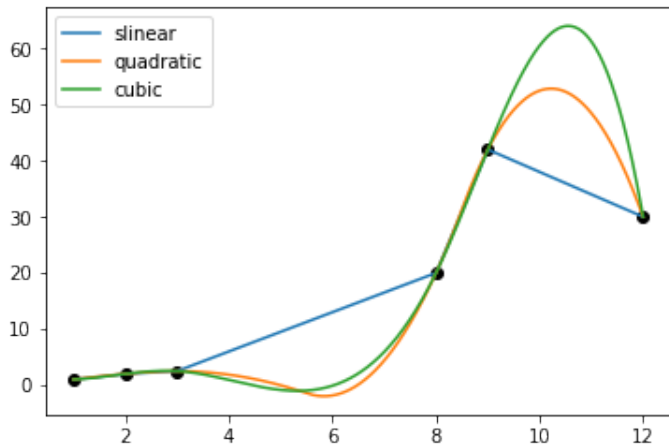
# создание сплайнов
f1 = interp1d(X,Y,'slinear')
f2 = interp1d(X,Y,'quadratic')
f3 = interp1d(X,Y,'cubic')

# набор точек для интерполяции
X0 = np.linspace(min(X), max(X), 1000)
Y1 = [f1(x) for x in X0]
Y2 = [f2(x) for x in X0]
Y3 = [f3(x) for x in X0]

plot(X,Y, 'o', color='black')
plot(X0, Y1, label='slinear')
plot(X0, Y2, label='quadratic')
plot(X0, Y3, label='cubic')

legend(loc='best')
show()
```

## Пример 2





## Пример 3

Интреполяция функции задающей поверхность

```
x = np.array([-5.0, -1.7, 1.7, 5.0])
y = np.array([-5.0, -1.7, 1.7, 5.0])
z = np.array([[ 50. , 27.8, 27.8, 50. ],
               [ 27.8, 5.6 , 5.6 , 27.8],
               [ 27.8, 5.6 , 5.6 , 27.8],
               [ 50. , 27.8, 27.8, 50. ]])

f = interp2d(x, y, z, kind='cubic')
```

Примеры использования модуля interpolate:  
[docs.scipy.org/doc/scipy-0.14.0/reference/tutorial/interpolate.html](https://docs.scipy.org/doc/scipy-0.14.0/reference/tutorial/interpolate.html)

# Outline

Введение

Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

Линейная алгебра

Полиномы

Численные методы

Минимизация функции

Интерполяция

Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

Ссылки и литература

# Outline

## Введение

## Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

## Линейная алгебра

## Полиномы

## Численные методы

Минимизация функции

Интерполяция

## Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

## Ссылки и литература

# Гистограмма

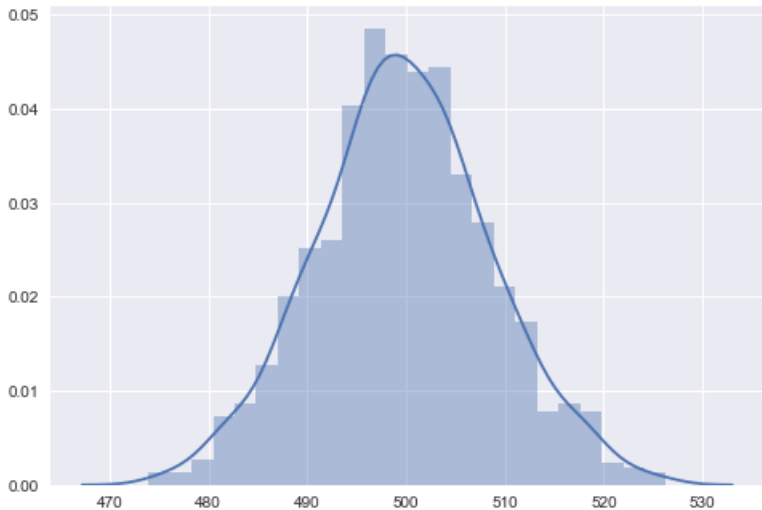
```
from random import random
import seaborn
from matplotlib.pyplot import show

X = [ sum( [random() for i in range(1000)] )
      for j in range(1000)]

# подготовим гистограмму и кривую распределения
seaborn.distplot(X)

show()
```

# Гистограмма



# Некоторые распределения

Распределения случайных величин

```
from scipy.stats import norm # нормальное распр.  
from scipy.stats import f # распределение Фишера  
from scipy.stats import t # распред. Стьюдента (t распред)  
from scipy.stats import chi2 # распред. Хи-квадрат  
from scipy.stats import poisson # распред. Пуассона
```

# Некоторые распределения

Работа с каждой из случайных величин может быть организована с помощью методов

- ▶ `rvs( size )` сгенерировать `size` случайных значений. может потребоваться указать параметры распределения.
- ▶ `pdf( x )` - Probability density function at `x`
- ▶ `cdf( x )` - Cumulative distribution function
- ▶ `isf( p )` - Inverse survival function



## Некоторые распределения

Вычислить вероятность того, что рост наугад выбранного человека будет меньше 180 см.

Средний рост людей 172 см.

Стандартное отклонение 7 см.

```
from scipy.stats import norm  
norm.cdf( (180 - 172) / 7 )  
0.87345104552644226
```

Сколько людей на Земле имеют рост меньше 180?

```
norm.cdf( (180 - 172) / 7 ) * 7.6e9  
6 638 227 946
```

# Outline

## Введение

## Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

## Линейная алгебра

## Полиномы

## Численные методы

Минимизация функции

Интерполяция

## Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

## Ссылки и литература

# Коэффициент корреляции

```
import seaborn  # для визуализации

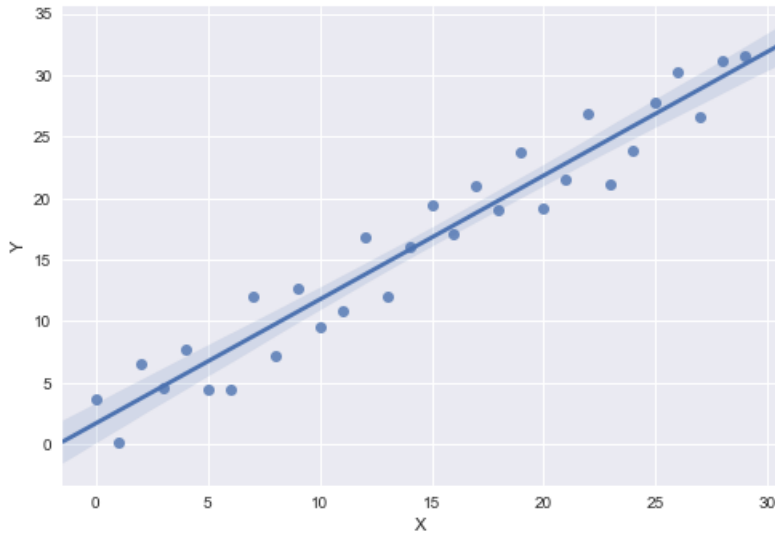
# Таблица для хранения стат.данных
from pandas import DataFrame

# поместим в таблицу как столбцы с заголовками X и Y
D = DataFrame( {'X':X, 'Y':Y} )

# Построим диаграмму рассеивания
seaborn.regplot(x='X', y='Y', data=D);

plt.show()
```

## Диаграмма рассеивания



# Outline

## Введение

## Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

## Линейная алгебра

## Полиномы

## Численные методы

Минимизация функции

Интерполяция

## Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

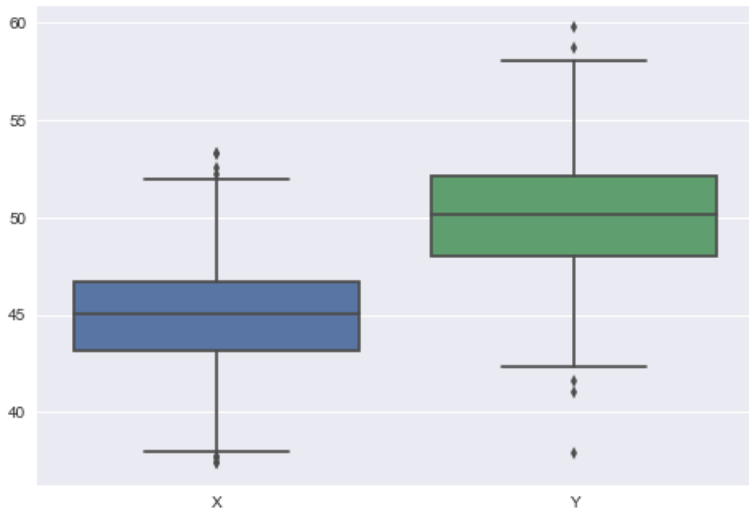
Диаграмма размаха

## Ссылки и литература

## Диаграмма размаха ("Ящик с усами")

```
X = [ sum([random() for i in range(90)])  
      for j in range(1000) ]  
Y = [ sum([random() for i in range(100)])  
      for j in range(1000) ]  
  
D = DataFrame( {'X' : X, 'Y':Y} )  
seaborn.boxplot(D)  
plt.show()
```

## Диаграмма размаха ("Ящик с усами")



О визуализации статистический данных и проверке гипотез с помощью Python:

[nahlogin.blogspot.ru/2016/01/pandas.html](http://nahlogin.blogspot.ru/2016/01/pandas.html)



# Outline

Введение

Диаграммы и графики

Векторизация функций

Двумерные графики

Графики в изометрии

Интерактивные графики

Линейная алгебра

Полиномы

Численные методы

Минимизация функции

Интерполяция

Теория вероятностей и статистика

Гистограмма

Корреляция и диаграмма рассеивания

Диаграмма размаха

Ссылки и литература

# Ссылки и литература

- ▶ Numerical methods in engineering with Python 3 / Jaan Kiusalaas.
- ▶ [try.jupyter.org](https://try.jupyter.org)
- ▶ Как использовать Jupyter (ipython-notebook) на 100%

# Ссылки и литература

Ссылка на слайды

[github.com/VetrovSV/Programming](https://github.com/VetrovSV/Programming)